

09/28/00

JC914 U.S. PTO

09-29-00

A

UTILITY
PATENT APPLICATION
TRANSMITTALAttorney Docket No. 40921-205583
First Named Inventor Manoj Nair
Title **Negative Cache Management System and Method**
Express Mail Label No. EL 507 833 081 US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Director for Patents
Box Patent Application
Washington, D.C. 20231

1. ☒ Specification Total Pages 18
(preferred arrangement as set forth below)
- Descriptive title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claim(s)
- Abstract of the Disclosure
2. ☒ Drawing(s) (35 USC 113) Total Sheets 6 (Figures 1-6)
3. Oath or Declaration Total Pages _____
a. ☒ Executed (original or copy)
b. ☐ Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with box 17 completed) [Note Box 4 below]
i. ☐ **DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, See 37 CFR 1.63(d)(2) and 1.33(b)
4. ☐ Incorporation By Reference (usable if Box 3b is checked)
The entire disclosure of the prior application, which a copy of the oath or declaration is supplied under Box 3b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein

5. ☐ Microfiche Computer Program (Appendix)
6. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
a. ☐ Computer Readable Copy
b. ☐ Paper Copy (identical to computer copy)
c. ☐ Statement verifying identify of above copies

ACCOMPANYING APPLICATION PARTS

7. ☒ Assignment Papers (cover sheet & document(s))
8. ☒ 37 CFR 3.73(b) Statement ☒ Power of Attorney (when there is an assignee)
9. ☐ English Translation Document (if applicable)
10. ☐ Information Disclosure ☐ Copies of IDS Statement (IDS)/PTO-1449 Citations
11. ☐ Preliminary Amendment
12. ☒ Return Receipt Postcard (MPEP 503)
13. ☐ Small Entity ☐ Statement filed in prior application, Status still proper and desired
14. ☐ Certified Copy of Priority Document(s) (If foreign priority is claimed)
15. Other:
☒ Check in the amount of \$
During the pendency of this application, the Commissioner is hereby authorized to credit overpayments or charge any additional fees under 37 CFR 1.16 and 1.17 to Deposit Account No 16-1435

16. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information below and in a preliminary amendment:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.:
Prior application information: Examiner: Group/Art Unit:

17. FEE CALCULATIONS

CLAIMS	For	Number Filed	Extra	Rate	Calculations
Total Claims	38	- 20 =	18 x	\$18 =	\$ 324.00
Indep. Claims	2	- 3 =	0 x	\$78 =	\$
Multiple Dependent Claims (if applicable)			+	\$260 =	\$
				Basic Fee (37 CFR 1.16)	\$ 690.00
				Total Calculations	\$
				Reduced by 50% for filing small entity (Note 37 CFR 1.9, 1.27, 1.28)	\$ -
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31) \$40 per property					\$ 40 00
TOTAL FEES SUBMITTED					\$1,054.00

18. CORRESPONDENCE ADDRESS

Name	A Jose Cortina				
	KILPATRICK STOCKTON LLP				
Address	3737 Glenswood Avenue				
	Suite 400				
City	Raleigh	State	NC	Zip Code	27612
Country	U.S.A.	Telephone	919.420.1700	Fax	919.420.1800
Name (Print/Type)	Bambi F. Walters			Registration No (Attorney/Agent)	45,197
Signature	<i>Bambi F Walters</i>			Date	September 26, 2000

EXPRESS MAIL CERTIFICATE

"Express Mail" mailing label number: EL 507833081 US


Date of Deposit: September 28, 2000

Type of Document(s): Utility Patent Application Transmittal Coversheet;
Executed Declaration and Power of Attorney;
Executed Assignment and Coversheet;
Eighteen (18) Sheets of Specification, Abstract, and
Claims;
Six (6) pages of drawings (Figures 1-6);
Check in the amount of \$1,054.00 and
Return postcard

Serial No.: Unassigned

Date Filed: Herewith

I hereby certify that the documents identified above are being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and are addressed to the Assistant Director for Patents, Washington, D.C. 20231.


Delesa Tillery

TITLE OF THE INVENTION

NEGATIVE CACHE MANAGEMENT SYSTEM AND METHOD

COPYRIGHT NOTIFICATION

5 A portion of the disclosure of this patent document and its figures contain material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyrights whatsoever.

BACKGROUND OF THE INVENTION

1. Field of the Invention

10 This invention relates to data processing systems. More particularly this invention provides a negative cache management system and method for controlling valid and invalid memory component identifiers, and, thereby, accelerates performance of relatively high speed
15 memory components, including hardware and software systems, that operate with storage devices.

2. Background

20 Data processing systems that are coupled to a large amount of data storage may spend significant amounts of CPU and I/O BUS time reading and writing data to and from data storage devices, such as disk drives or similar longer term data storage devices. Accordingly, many data processing systems include relatively high speed, short term caching systems that can be accessed much more rapidly by a coupled computer system without the delays associated with the mechanical motion and time delay inherent in a long term storage device. Conventional data processing systems may include caches at several levels of operation, such
25 as, cache memory, disk cache, RAM, "macro" caches, local server caches, and web browser caches.

One of the most well known caching systems is the use of a name cache to access a directory or a file stored through the use of tree structured directories. Each disk contains a root directory, one or more subdirectories, and one or more files. Each directory contains entries for additional directories and files. A named file may be at the end of a path that passes through a root directory and one or more subdirectories (i.e., /Directory/Subdirectory/Filename). A specific file may be identified by accessing the drive, path, and filename. When a user requests a specific file, the operating system file system must take the file's pathname and search through all of the components (i.e., directories and subdirectories) to identify the location of the file associated with that filename.

In conventional file systems, the file system typically maintains a name cache containing the location of files that have earlier been requested and found. This avoids having to search again for a file every time that file is accessed. Traditionally, the unified file manager (UFM) of the file system receives a pathname/filename and first looks in the name cache to see if the location of the requested file identifier is already there. If not, the file identifier is passed to the flat file manager (FFM) that starts searching for the file in the directory. Some users might have thousands, tens of thousands, or even conceivably more files in a single directory, making it potentially very time consuming to search.

If an invalid file identifier is received, the file system will search through all directories in the PATH environment variable without finding it. Since the requested file identifier was not successfully located, nothing gets placed in the name cache. If users frequently ask for the same invalid file identifiers, the file system will be forced to repeatedly do useless searching that takes up valuable and CPU cycles and disk I/O. Accordingly, what is needed is a system and method for accelerating performance of relatively high speed memory components in such a way as to ensure that the memory addresses of invalid memory component identifiers are managed by a cache system and method that may be utilized by a coupled host computer in a relatively short period of time.

SUMMARY OF THE INVENTION

This invention provides an intelligent caching system and method that immediately accelerate the response time and throughput of all existing relatively high speed memory components, such as storage systems that include disk drives and arrays, connected to either single-, dual-, or multi- host computers. To achieve this, the caching system and method maintain one or more caches in memory that manage valid and invalid memory component identifiers, such as, caches of recently searched valid and invalid file and object identifiers, including directory files, as well as other miscellaneous information, that are used to locate directories and files in storage devices.

The caching system and method are transparent to the host computer and have an easily configurable cache capacity. In an embodiment, a negative caching system and method manages only the invalid directories and files. The negative cache management system and method can be configured as a predetermined percentage of capacity, such as 1% of a caching system and method populated with valid memory component identifiers. Alternatively, a predetermined finite number of spaces can be set aside for the negative cache. A separate negative cache is added to keep invalid memory component identifiers. This negative name cache, like the name cache, searches for a memory component identifier, such as a specified filename, when a memory component identifier is first received. If the name of the memory component identifier is stored in the negative name cache, the operating system knows that no further searching is required.

As an alternative, invalid memory component identifiers could be entered into the same name cache as the valid memory component identifiers. In this single cache embodiment, valid and invalid memory component identifiers can be differentiated by adding a validity field to the cache entry or, alternatively, by placing a default entry indicating invalidity in the field that would normally contain the location of the data fields. Similar to the embodiment that utilizes a separate negative cache, this single cache could be partitioned

and configured to store a predetermined percentage of valid, such as 99% of capacity, and a predetermined percentage, such as 1% of capacity, of invalid memory component identifiers. Alternatively, a predetermined finite number of spaces can be set aside for the invalid memory component identifiers.

5

In another embodiment, this invention includes (a) logic to check the cache to determine if a received memory component identifier is invalid, (b) logic to add the invalid memory component identifier to the cache when a search by the flat file manager is unsuccessful, (c) logic to monitor new memory component identifiers being created and to
10 remove an entry from the invalid list if that path name later becomes valid, (d) logic to monitor memory component identifiers being deleted or moved and to add an entry from the invalid list if that path name becomes invalid, (e) logic to manage the most frequently used invalid memory component identifiers, and (f) logic to manage the most recently used invalid memory component identifiers.

006724.09300

15

This invention is useful in providing a system and method for rapidly identifying an invalid memory component identifier, such as a directory/filename, after such memory component identifier is initially searched for and determined to be invalid.

20

Another use of this invention is to provide a system and method for rapidly identifying an invalid memory component identifier, such as a directory/filename, after such memory component identifier is deleted from or moved to another location of the storage device.

25

Another use of this invention is to reduce the time required to locate an invalid memory component identifier by accessing a negative cache that manages invalid memory component identifiers, and, therefore, eliminates the necessity of searching through the directories and filenames of a storage device, such as a disk drive.

Another use is to provide a system and method for rapidly identifying invalid memory component identifiers that have been recently searched for and that are frequently searched for. This is achieved by creating and maintaining an intelligent negative cache that uses the history of searches for invalid memory component identifiers.

5

These uses may be accomplished singularly, or in combination, in one or more of the embodiments of the present invention.

Additional uses, objects, advantages, and novel features of the invention is set forth in the detailed description that follows and becomes more apparent to those skilled in the art upon examination of the following or upon learning by practice of the invention.

BRIEF DESCRIPTION OF THE FIGURES

Other objects, advantages, and novel features of the present invention are more clearly understood by reference to the following description taken in connection with the accompanying figures, in which:

FIG. 1 is a representation of a data storage directory used in data processing systems.

FIG. 2 is a flowchart illustrating PATH usage in data processing systems.

FIG. 3 is a prior art representation of the memory components of a data processing system.

FIG. 4 is a flowchart illustrating the prior art method of searching for a filename in a data processing system.

FIG. 5 is a representation of the memory components of a data processing system including use of a negative name cache in an embodiment of the present invention.

FIG. 6 is a flowchart illustrating the method of searching for a memory component identifier in a data processing system that includes use of a negative name cache in an embodiment of the present invention.

DETAILED DESCRIPTION

Referring now in detail to an embodiment of the invention, an example of which is illustrated in the accompanying attachments, the present invention makes use of two basic concepts. One concept is creating a cache system and method that contain valid and invalid memory component identifiers. For example, in response to an initial request to locate a specified filename, an operating system searches for data stored in a directory structure of a storage device, such as a disk drive. If the specified filename is successfully located, it is added to a name cache in main memory. If the specified filename is not successfully located, it is added to a negative name cache in main memory. Thus, the present invention is different from conventional caching systems and methods in that it creates a negative cache for memory component identifiers that are not actually located on the storage device.

The second concept manages the negative cache based on the memory components coupled to the host computer. For example, in a disk caching system, a negative name cache is maintained with the unified file manager. The next time a search is requested for the specified file identifier, the parent directory and the filename stored in the negative name cache can be immediately accessed by a unified file manager to determine whether the specified filename is stored on the disk. Thus, the present invention provides a negative cache management system and method that can be used to accelerate the input/output performance of various relatively high speed memory components, such as disk drives, arrays, routers, and buffers, connected to either single-, dual-, or multi- host computers.

Turning now to the attached figures, Fig. 1 depicts a conventional file system that is rooted at directory called "/" 100 and is referred to as the root directory. The root directory 100 has subdirectories "usr" 110, "tmp" 120, and "etc" 130 branching off of it. Each of these subdirectories directories 110-130 contains files 121-122 or other subdirectories 111, 131. The UNIX "ls" command executable file 112 resides at "/usr/bin" 111 and its pathname is referred to as "/usr/bin/ls" 112. When a user types the command "ls," the system consults a special environment variable that instructs the system where "ls" could possibly be located. That variable is called a PATH environment variable, and an example of a PATH environment variable is provided below.

PATH= /tmp : /etc/bin : /usr/bin

This PATH statement tells the system that whenever an executable file is searched for, the system should look in "/tmp" 120 first. If not found in "/tmp" 120 then look in "/etc/bin" 131 next and finally, if still not found, look in "/usr/bin" 111. As it turns out in our scenario, "ls" happens to reside in the last directory, "/usr/bin" 111. This algorithm is sketched in Figure 2.

Fig. 2 illustrates the flowchart of PATH usage. When a request is received to execute command "ls," 201, the system consults the PATH environment variable 202 and gets the next component of the PATH 203. Thereafter, the system concatenates the component of PATH to "ls," 204 and looks up "ls" in the file system 205. The system queries if the file is found 206. If so, then the system executes the command 207. However, if the file is not found, the system gets the next component of PATH 203 until the file is found.

The problem with this system is that if "ls" is a frequently invoked command that is not found in the "/tmp" directory, then the system is going to be frequently searching for it and wasting time. The system is always going to start the search at "/tmp" and this is always

going to fail. In some systems, the “/tmp” directory contains many thousands of files and searching through such a huge directory is unacceptably slow.

Figs. 3 and 4 depicts a conventional UFM subsystem that receives a directory pointer and a file name to look for in that directory. In this illustrative example, the directory is “/tmp” and the file name is “ls.” To speed such frequent searches, the UFM 302 uses a Name Cache 303 that contains recent successful search results. However, an entry for “/tmp/ls” is not going to exist in the name cache because no such file exists in “/tmp.” The Name Cache 303 remembers only successful results, not failures. In the case of frequent failures, the Name Cache 303 is of no help, and the Name Cache 303 actually decelerates the processing system because it spends CPU cycles trying to locate something that does not exist. After failing to find “/tmp/ls” in the Name Cache 303, the system moves on and ask the FFM 305 to go look for “/tmp/ls.” FFM 305 asks the Buffer Manager (BM) 306 to provide a buffer containing a disk copy of the directory contents. This step may involve disk Input/Output (I/O) 308. The FFM 305 then traverses the directory contents one file at a time until the file is found or the end of the directory has been reached. This step involves accessing the Block Special File Manager (BSFM) 307 and accessing disk Input/Output 308. In the case of a huge “/tmp” directory, the system must inspect each one of its thousands of files before all entries are exhausted to declare failure. The problem with this search is that the system doesn’t remember the results of that painful computation and it throws it away. The next time the “ls” command is invoked, the system repeats that expensive failure sequence of computations. Figure 4 is a flowchart describing the above algorithm and illustrates how conventional name caches are populated.

Figs. 5 and 6 depict the UFM subsystem of the present invention including the negative name cache that receives a directory pointer and a file name to look for in that directory. In this illustrative embodiment, the directory is “/tmp” and the file name is “ls.” To speed such frequent searches, the UFM 302 uses a both a Name Cache 303 that contains recent successful search results and a Negative Name Cache 503 that contains recent

unsuccessful search results. In this second illustrative embodiment, an entry for “/tmp/ls” is going to exist in the Negative Name Cache 503 because no such file exists in “/tmp” as established by a directory/filename search for “tmp/ls.” The Negative Name Cache 503 remembers the failed attempt to locate an entry for “/tmp/ls,” and the system accelerates its CPU time by immediately identifying the invalid search request and avoiding the repeated computation used by conventional systems. Figure 6 is a flowchart describing the above algorithm and illustrate how an embodiment of the negative name cache is populated.

An example of the advantages of the negative name caching system is shown in Table 1 below for an exemplary storage device with two processors and a fileset that consists of 50 file systems.

	CONVENTIONAL CACHING SYSTEM	NEGATIVE CACHING SYSTEM
Files	1950050	1950050
Files accessed for I/O operations	195000	195000
Directories	65000	65000
Filename lookups/second when 3% of files are invalid	4000	5000

The above files are spread over the 50 file systems with a random attempt to lookup filename in the system where 3% of the named files are invalid. The above results show improved CPU and I/O performance of 25%.

The present invention may be implemented in numerous embodiments. In one embodiment, a separate negative name cache is added to keep invalid file identifiers (e.g., Directory/Subdirectory/InvalidFilename). This cache, like the name cache, is searched when a file identifier is first received. If the name appears in the negative name cache, the operating system knows that no further searching is required. This prevent the need to search any further and results in a very significant performance boost.

In another embodiment, invalid file identifiers could be entered into the same name cache as the valid file identifiers. In this single cache implementation, valid and invalid file identifiers could be differentiated by inspecting the inode pointer embedded in the cache entry. The inode pointer is a file system structure that contains information about a single file. If the inode pointer is NULL, then the system identifies the file identifier as an invalid file that does not reside in the given directory. If the inode pointer is valid, then the system operates like a conventional name cache entry representing a success.

In another embodiment, this invention includes (a) logic to check the cache to determine if a received memory component identifier is invalid, (b) logic to add the invalid memory component identifier to the cache, such as adding a file identifier to the cache when a search in a directory for a file name is unsuccessful, (c) logic to monitor new memory component identifiers being created and to remove an entry from the invalid list if that file identifier later becomes valid, (d) logic to monitor memory component identifiers being deleted or moved and to add an entry from the invalid list if that file identifier becomes invalid, (e) logic to manage the most frequently used invalid memory component identifiers, and (f) logic to manage the most recently used invalid memory component identifiers.

In another embodiment, the negative cache size is variable, as it could have different size requirements. For example, the size of the negative cache may be based on the usage of the system or based on a predetermined percentage of capacity, such as 1% of the conventional caching system and method populated with valid memory component identifiers. Another example is to set aside a predetermined finite number of spaces for the negative cache. In addition, the negative cache may be initialized at the same time as the conventional cache during system initialization (i.e., boot time).

In another embodiment, the negative cache can be implemented by a number of standard cache implementation techniques. For example, when the name cache is constructed it initially contains no entries, positive nor negative. Thereafter, when a file

lookup operation results in failure, FFM will report back to UFM about the failure. UFM, in turn, will create a new negative file identifier and place it as an entry to the negative name cache. This file identifier will represent the fact that a particular file does not reside in a particular directory. Subsequent searches for this same pair (e.g., Directory/Filename) are quickly satisfied from the negative name cache. When the negative name cache fills up, the present invention in an embodiment uses a random replacement policy in which a random entry in the cache is thrown out to make room for the new entry just created. Additionally, when a file is created, UFM needs to remove a corresponding entry from the negative name cache if such an entry exists.

The foregoing description and associated figures detail only illustrative examples of the environment in which the invention can be used and are not intended to be limiting. In addition, various programming languages, software platforms, operating systems, hardware components, memory components, and other technology may utilize the negative cache system and method to provide more efficient and reliable means for accessing data in storage devices. Furthermore, current technology can always be enhanced to incorporate the most advanced available technology to implement the negative cache system and method. Variations and modifications of the present invention is apparent to one skilled in the art, and the above disclosure is intended to cover all such modifications and equivalents.

What is claimed is:

1. A caching system for identifying memory component identifiers associated with data in a storage device, comprising:

means for creating a cache of said memory component identifiers, wherein

said memory component identifiers comprise identifiers that are invalid; and

means for managing said cache of memory component identifiers.

2. The caching system of claim 1, wherein said memory component identifiers further comprise one or more of the following:

identifiers that are moved from a first storage location to a second storage

location;

identifiers that are deleted;

identifiers that are dynamic;

identifiers that are renamed; and

identifiers selected by a user.

3. The caching system of claim 1, wherein said memory component identifiers further comprise identifiers that are valid.

4. The caching system of claim 1, wherein said storage device is a data processing system comprising:

a disk drive including a disk in which is stored a tree structure of data located in directories and files; and

a main memory for storing data, said data stored in said memory being accessible at a rate substantially faster than the rate at which data stored on a disk can be accessed.

5. The caching system of claim 4, wherein said means for creating said cache comprises:

creating in said main memory a first name cache;

in response to an initial request to open a specified file nested in a path of one or more directories, accessing said disk and determining that directory entries and file entries do not contain said specified file;

storing in said first name cache a history of paths of said directory entries

and said file entries that do not contain said specified file;

in response to a subsequent request to open said specified file, searching through said history of said first name cache to locate said directory entries and said file entries that do not contain said specified file; and

returning a response that said specified file is not contained on said disk.

6. The caching system of claim 1, further comprising:

means for updating said cache by removing a least recently used memory component identifier in accordance with a least recently used routine.

7. The caching system of claim 1, further comprising:

means for updating said cache by adding a most recently used memory component identifier in accordance with a most recently used routine.

8. The caching system of claim 1, further comprising:

means for updating said cache by adding a most frequently searched memory component identifier in accordance with a most frequently searched routine.

9. The caching system of claim 1, further comprising:

means for updating said cache by removing a least frequently searched memory component identifier in accordance with a least frequently searched routine.

10. The caching system of claim 1, further comprising:

means for updating said cache by updating said memory component identifier in accordance with a user specified routine.

11. The caching system of claim 1, wherein said cache is one of a negative cache of memory component identifiers that are not associated with data in said storage device.

12. The caching system of claim 11, wherein said negative cache comprises a predetermined number of cache entries for storing a history of said memory component identifiers that are not associated with data in said storage device.

13. The caching system of claim 12, wherein said predetermined number of cache entries is based on usage of said memory component identifier.

14. The caching system of claim 11, wherein said negative cache comprises a percentage of cache entries stored in a cache system of valid memory component identifiers.
15. The system of claim 14, wherein said negative cache is used for storing a history of said memory component identifiers that are not associated with data in said storage device.
- 5 16. The caching system of claim 11, wherein the cache is further comprises a positive cache of memory component identifiers that have been written to at least one storage device.
17. The caching system of claim 1, further comprising:
means for determining if at least one memory component identifier has
been accessed by at least one of a plurality of computer processes.
- 10 18. The caching system of claim 17, further comprising:
means for increasing the number of said memory component identifiers to
be placed into said cache if at least one memory component identifier has been
accessed by at least one of said plurality of computer processes.
- 15 19. The caching system of claim 17, further comprising:
means for decreasing the number of said memory component identifiers to
be placed into said cache if at least one memory component identifier has been
accessed by at least one of said plurality of computer processes.
20. A caching method for identifying memory component identifiers associated with data in
a storage device, comprising:
creating a cache of said memory component identifiers, wherein said
memory component identifiers comprise identifiers that are invalid; and
managing said cache of memory component identifiers.
21. The caching method of claim 20, wherein said memory component identifiers further
comprise one or more of the following:
25 identifiers that are moved from a first storage location to a second storage
location;
identifiers that are deleted;
identifiers that are dynamic;
identifiers that are renamed; and

identifiers selected by a user.

22. The caching method of claim 20, wherein said memory component identifiers further comprise identifiers that are valid.

23. The caching method of claim 20, wherein said storage device is a data processing system comprising:

a disk drive including a disk in which is stored a tree structure of data located in directories and files; and

a main memory for storing data, said data stored in said memory being accessible at a rate substantially faster than the rate at which data stored on a disk can be accessed.

24. The caching method of claim 23, wherein creating said cache comprises:

creating in said main memory a first name cache;

in response to an initial request to open a specified file nested in a path of one or more directories, accessing said disk and determining that directory entries and file entries do not contain said specified file;

storing in said first name cache a history of paths of said directory entries and said file entries that do not contain said specified file;

in response to a subsequent request to open said specified file, searching through said history of said first name cache to locate said directory entries and said file entries that do not contain said specified file; and

returning a response that said specified file is not contained on said disk.

25. The caching method of claim 20, further comprising:

updating said cache by removing a least recently used memory component identifier in accordance with a least recently used routine.

26. The caching method of claim 20, further comprising:

updating said cache by adding a most recently used memory component identifier in accordance with a most recently used routine.

27. The caching method of claim 20, further comprising:

updating said cache by adding a most frequently searched memory

component identifier in accordance with a most frequently searched routine.

28. The caching method of claim 20, further comprising:

updating said cache by removing a least frequently searched memory component identifier in accordance with a least frequently searched routine.

5 29. The caching method of claim 20, further comprising:

updating said cache by updating said memory component identifier in accordance with a user specified routine.

30. The caching method of claim 20, wherein said cache is one of a negative cache of memory component identifiers that are not associated with data in said storage device.

10 31. The caching method of claim 30, wherein said negative cache comprises a predetermined number of cache entries for storing a history of said memory component identifiers that are not associated with data in said storage device.

32. The caching method of claim 31, wherein said predetermined number of cache entries is based on usage of said memory component identifier.

15 33. The caching method of claim 30, wherein said negative cache comprises a percentage of cache entries stored in a cache system of valid memory component identifiers.

34. The system of claim 33, wherein said negative cache is used for storing a history of said memory component identifiers that are not associated with data in said storage device.

20 35. The caching method of claim 30, wherein the cache is further comprises a positive cache of memory component identifiers that have been written to at least one storage device.

36. The caching method of claim 20, further comprising:

determining if at least one memory component identifier has been accessed by at least one of a plurality of computer processes.

37. The caching method of claim 36, further comprising:

25 increasing the number of said memory component identifiers to be placed into said cache if at least one memory component identifier has been accessed by at least one of said plurality of computer processes.

38. The caching method of claim 36, further comprising:

decreasing the number of said memory component identifiers to be placed into said cache if at least one memory component identifier has been accessed by at least one of said plurality of computer processes.

Table 1. Demographic characteristics of the study population	
Age (years)	65.0 ± 1.5
Gender	
Male	50.0
Female	50.0
Education (years)	12.0 ± 1.0
Marital status	
Married	60.0
Single	40.0
Occupation	
Retired	70.0
Unemployed	30.0
Income (USD/month)	1,200.0 ± 200.0
Health status	
Good	60.0
Poor	40.0
Comorbidities	
Hypertension	45.0
Diabetes	35.0
Cholesterol	40.0
Smoking status	
Smoker	20.0
Non-smoker	80.0
Alcohol consumption	
Regular	10.0
Occasional	30.0
Never	60.0

ABSTRACT

This invention provides a negative cache management system and method for controlling valid and invalid memory component identifiers, and, thereby, accelerates performance of relatively high speed memory components, including hardware and software systems, that operate with storage devices. This invention includes (a) logic to check the cache to determine if a received memory component identifier is invalid, (b) logic to add the invalid memory component identifier to the cache, such as adding a file identifier to the cache when a search in a directory for a file name is unsuccessful, (c) logic to monitor new memory component identifiers being created and to remove an entry from the invalid list if that file identifier later becomes valid, (d) logic to monitor memory component identifiers being deleted or moved and to add an entry from the invalid list if that file identifier becomes invalid, (e) logic to manage the most frequently used invalid memory component identifiers, and (f) logic to manage the most recently used invalid memory component identifiers.

WINLIB01:848365.1

Directory Tree

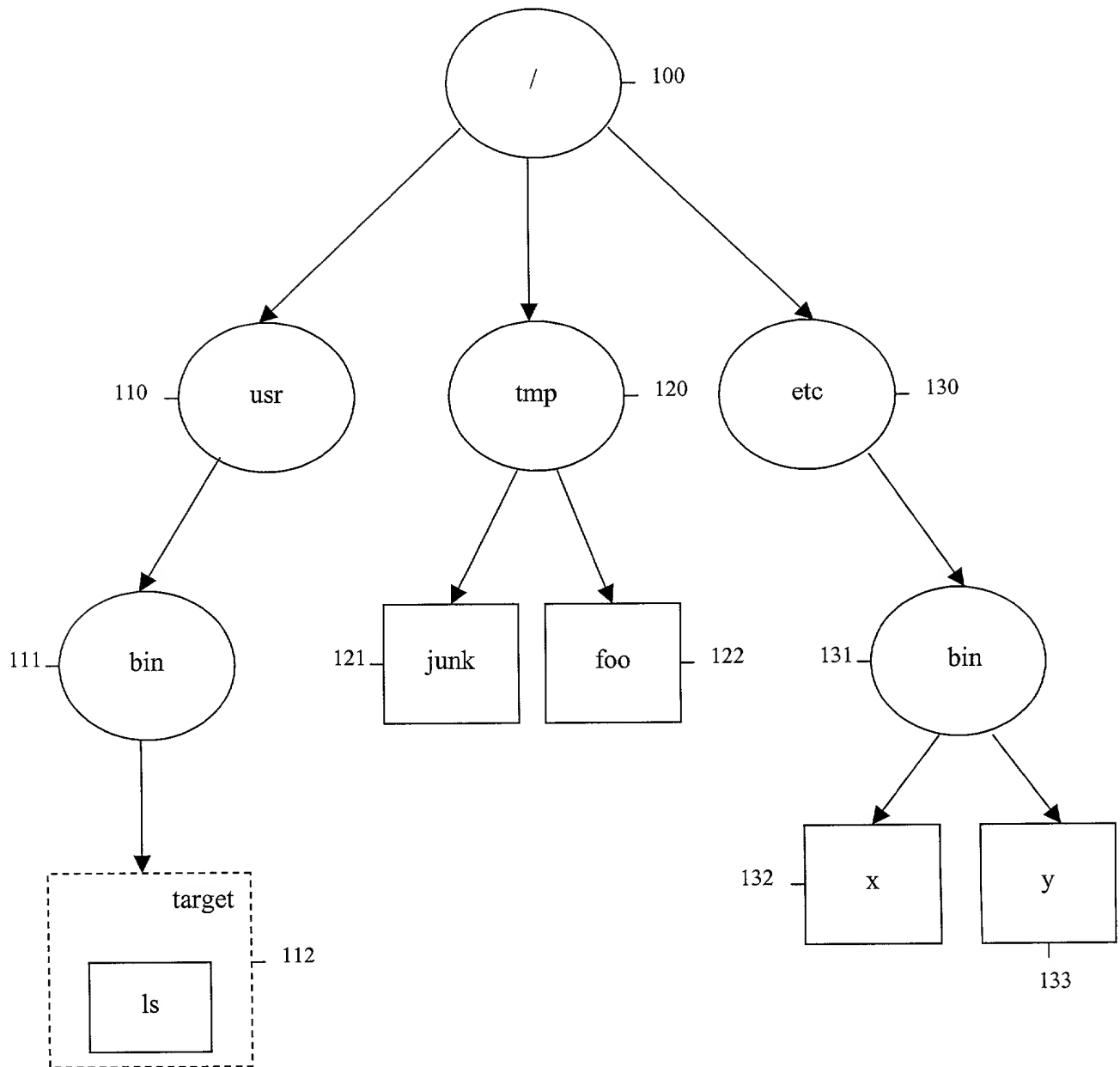


FIG. 1

Flowchart of PATH Usage

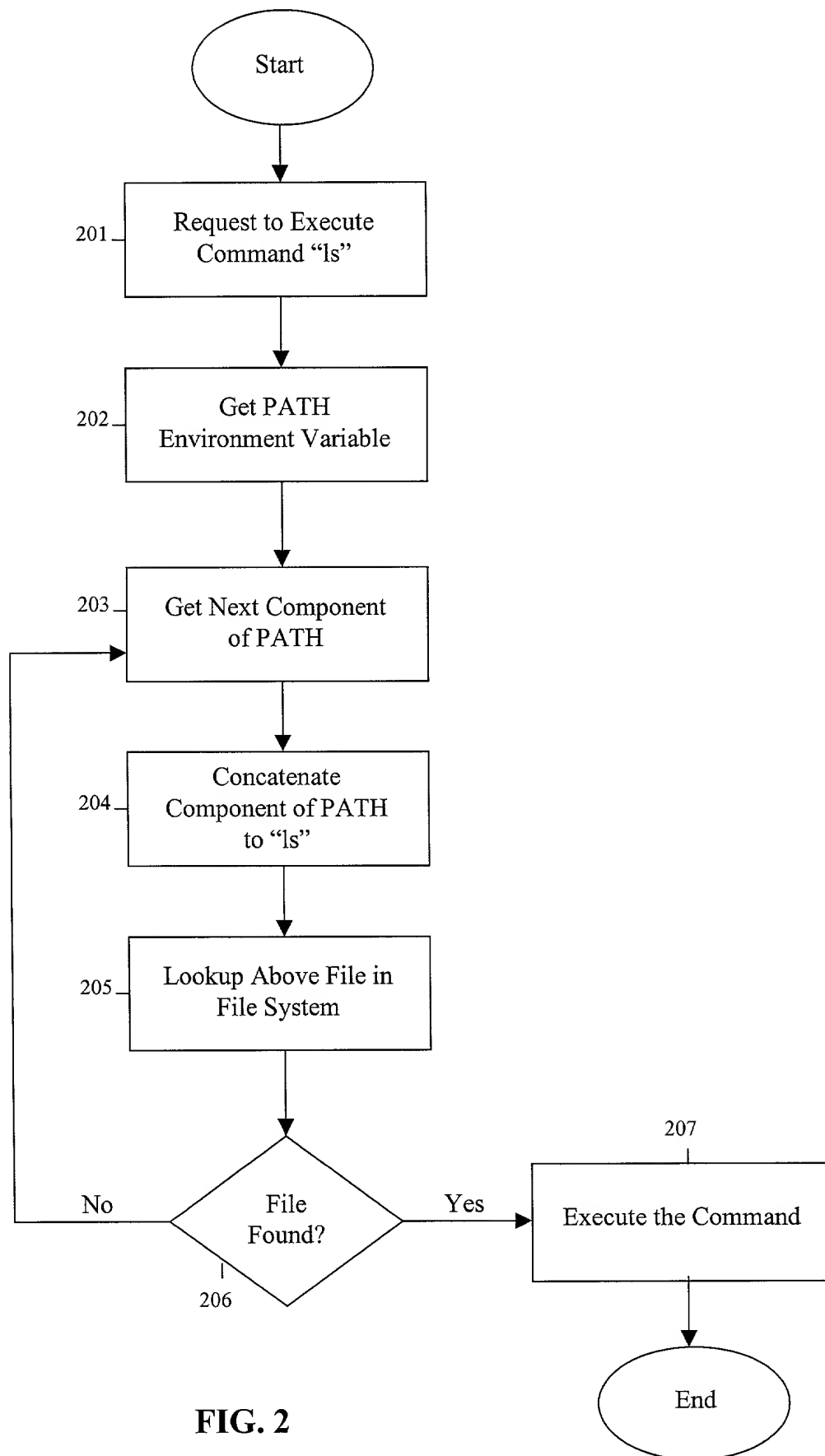


FIG. 2

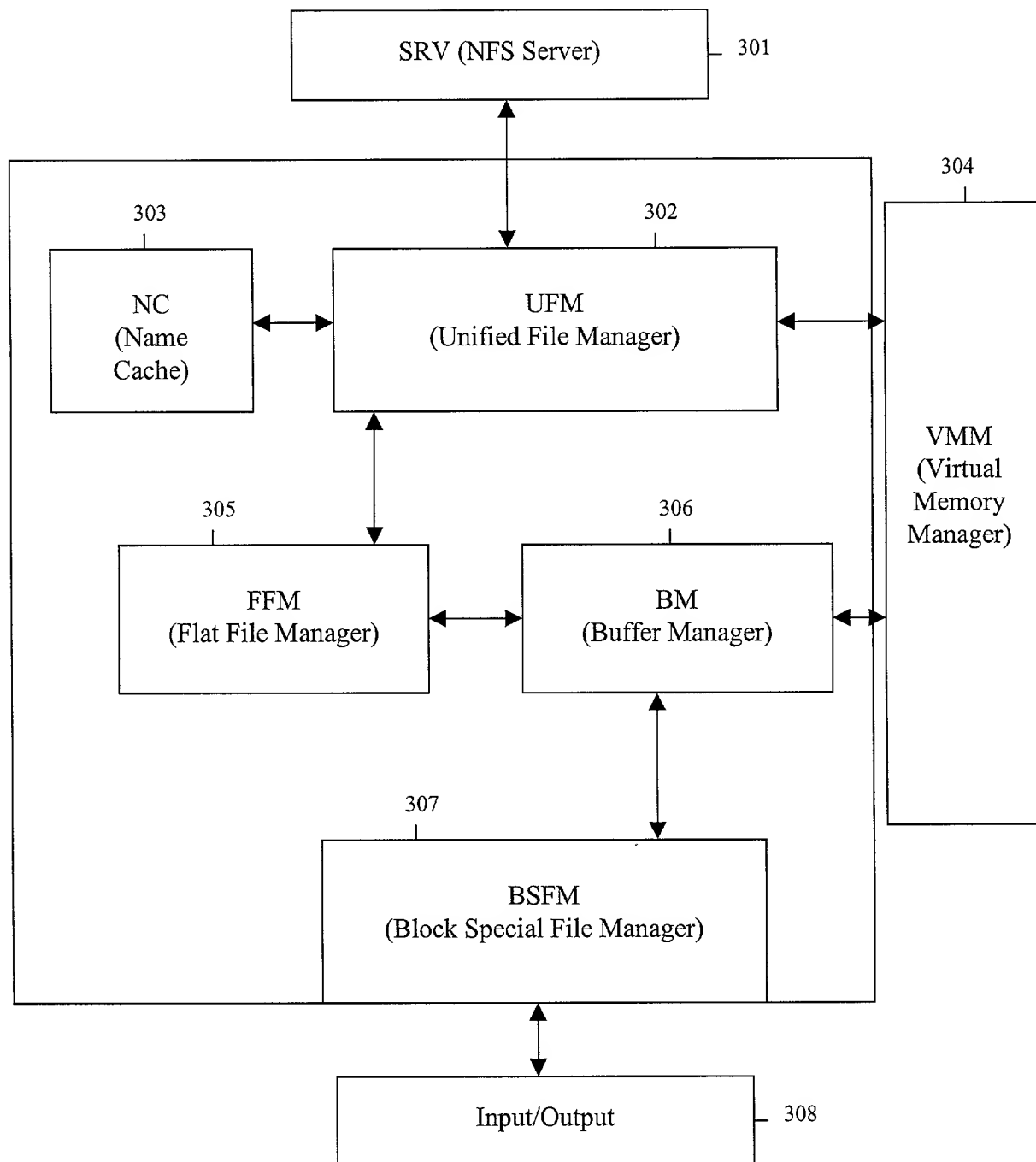


FIG. 3
(PRIOR ART)

Flowchart of File Lookup

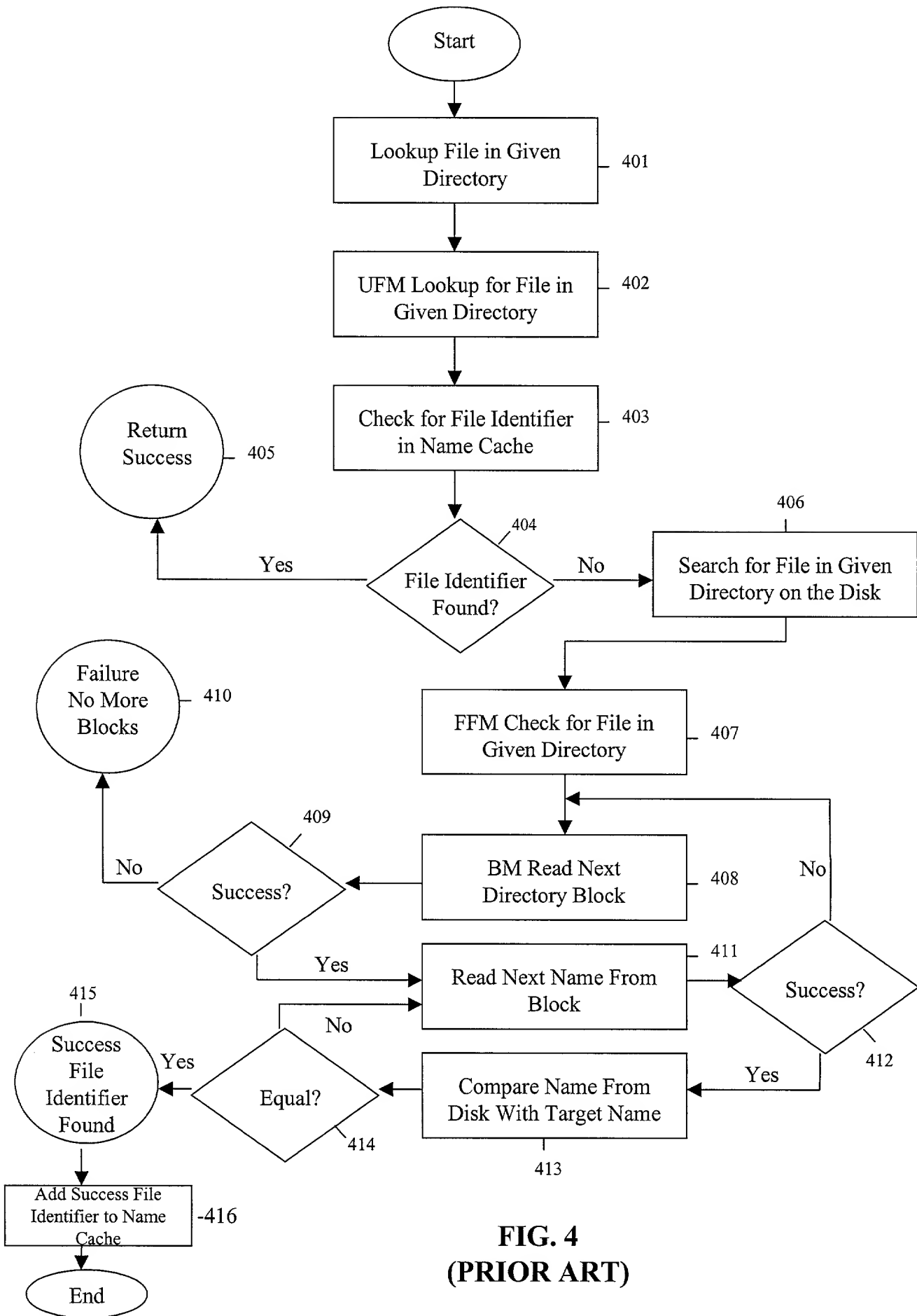


FIG. 4
(PRIOR ART)

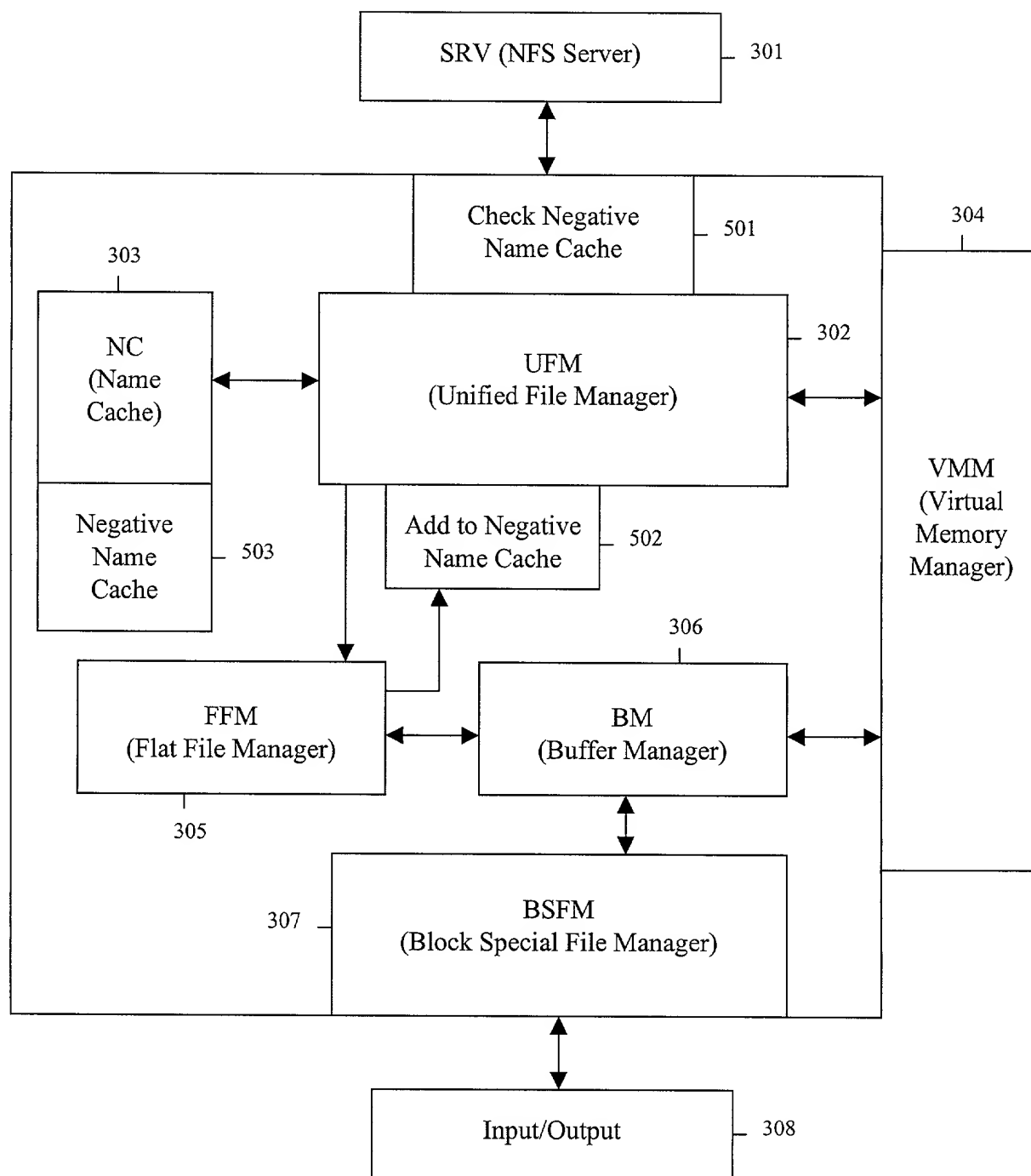


FIG. 5

Flowchart of File Lookup With a Negative Name Cache (NNC)

003250 " 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

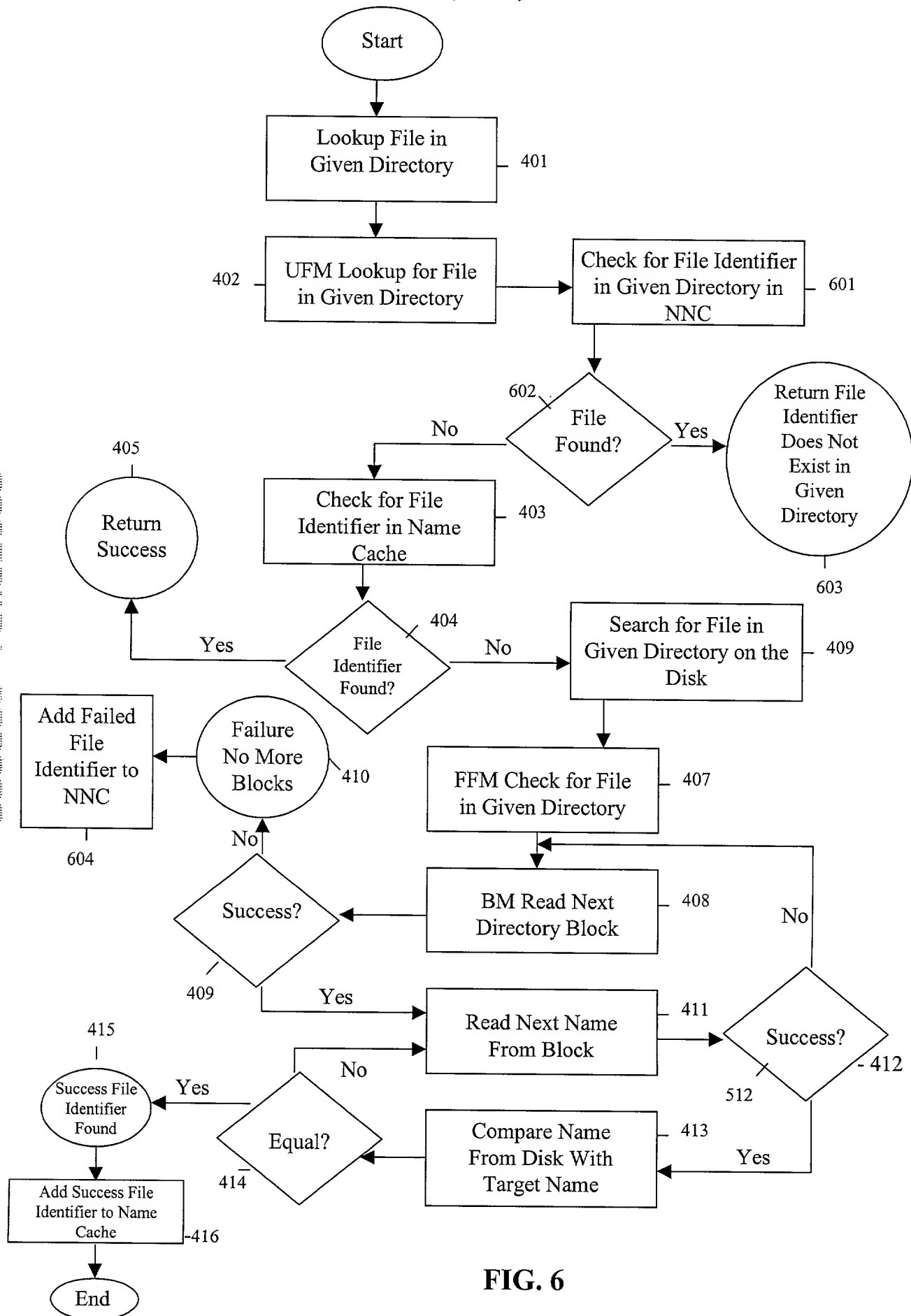


FIG. 6

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION
English Language Declaration

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled **Negative Cache Management System and Method**; the specification of which: (check one)

 X is attached hereto.

_____ was filed on _____ as _____

____ Application Serial No. _____

_____ and was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent of inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

Priority Claimed

NONE	
(Number)	(Country)

(Day/Month/Year Filed)

<u>Yes</u>	<u>No</u>
------------	-----------

(Number)	(Country)
1	USA
2	USA
3	USA
4	USA
5	USA
6	USA
7	USA
8	USA
9	USA
10	USA
11	USA
12	USA
13	USA
14	USA
15	USA
16	USA
17	USA
18	USA
19	USA
20	USA
21	USA
22	USA
23	USA
24	USA
25	USA
26	USA
27	USA
28	USA
29	USA
30	USA
31	USA
32	USA
33	USA
34	USA
35	USA
36	USA
37	USA
38	USA
39	USA
40	USA
41	USA
42	USA
43	USA
44	USA
45	USA
46	USA
47	USA
48	USA
49	USA
50	USA
51	USA
52	USA
53	USA
54	USA
55	USA
56	USA
57	USA
58	USA
59	USA
60	USA
61	USA
62	USA
63	USA
64	USA
65	USA
66	USA
67	USA
68	USA
69	USA
70	USA
71	USA
72	USA
73	USA
74	USA
75	USA
76	USA
77	USA
78	USA
79	USA
80	USA
81	USA
82	USA
83	USA
84	USA
85	USA
86	USA
87	USA
88	USA
89	USA
90	USA
91	USA
92	USA
93	USA
94	USA
95	USA
96	USA
97	USA
98	USA
99	USA
100	USA

(Day/Month/Year Filed)

Yes	No
-----	----

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

NONE
(Application Serial No.)

(Filing Date)

(Status)
(patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

English Language Declaration

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

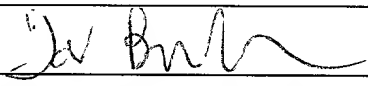

A. José Cortina, Registration No. 29,733; George T. Marcou, Registration No. 33,014; Benjamin Driscoll, Registration No. 41,571, Charles W. Calkins, Registration No. 31,814, John M. Harrington, Registration No. 25,592, Dawn-Marie Bey, Registration No. 44,442, John Ball, Registration No. 44,433; James J. Bindseil, Registration No. 42, 326; J. Steven Gardner, Registration No. 41,772; Bambi F. Walters, 45,197, and Jason Link, Registration No. 44,874.

Send Correspondence to:

Direct telephone calls to:

A. José Cortina
Kilpatrick Stockton LLP
3737 Glenwood Avenue, Suite 400
Raleigh, North Carolina 27612

A. José Cortina
(919) 420-1820

Full name of first inventor: Dov Bulka	
Inventor's Signature 	Date: September <u>27</u> , 2000
Residence: 7819 Falcon Rest Circle, Raleigh NC 27615	
Citizenship: United States of America	
Post Office Address: 7819 Falcon Rest Circle, Raleigh NC 27615	
Full name of second inventor: Manoj Nair	
Inventor's Signature 	Date: September <u>27</u> , 2000
Residence: 101 Kindletree Ct. Apt 2F, Cary, NC 27513	
Citizenship: India	
Post Office Address: 101 Kindletree Ct. Apt 2F, Cary, NC 27513	